Procesiranje formi

U prethodnim lekcijama smo se bavili samo GET zahtevima – situacijama gde korisnik dobavlja nešto sa servera. U praksi ćemo često biti u situaciji da želimo da damo mogućnost korisniku da nam pošalje podatke. Često je to slučaj sa komentarima, formama za logovanje (prijavljivanje) ili slanjem poruka. U ovim slučajevima koristimo HTTP metodu POST.

Same forme su zapravo deo HTTP protokola, na kojima je Djangov metod rada sa njima i baziran. HTML tagovi koji nam služe za klijentsko slanje podataka serveru su form, input i select. Tag form je zapravo kontejnerski objekat za sve podatke koje klijent želi da pošalje u tom trenutku. Sadrži dva bitna atributa – action i method:

- Atribut action se može nazvati i destinacijom ili linkom. Naime, vrednost ovog atributa je link na koji klijentski zahtev, nakon popunjavanja te forme, treba da bude poslat. Ako ovaj atribut u formi nije iskorišćen, taj zahtev će otići na link na kojem se klijent trenutno i nalazi.
- Atribut method nam govori koju HTTP metodu taj klijentski zahtev koristi. Pa tako, sa action atributom znamo gde se zahtev šalje, a preko atributa method – kako će taj zahtev biti poslat.

Recimo da svojoj Django aplikaciji želimo da dodamo funkcionalnost dodavanja nove knjige i godine njenog izdanja. Način postizanja toga korišćenjem HTML koda je sledeći:

HTML kod forme:

```
<form method="POST" action="/book-added/">
    <input type="text" name="book_title">
    <input type="text" name="book_year">
    <button type="submit">Add book!</button>
</form>
```

U ovom HTML primeru forma ima dva polja: book_title i book_year. Oba podatka se nakon klika na dugme *button* šalju na link /book-added/ (action atribut) metodom POST (method atribut). Takođe, POST metoda ove podatke šalje u telu poruke HTTP zahteva, što znači da je, ako je sajt na HTTPS-u, ovo polje zahteva enkripovatno. Da smo umesto POST metode imali GET, ovi podaci bi se poslali kao deo linka (query strings).

Atribut type taga input nam određuje kog tipa je ovaj HTML element. U ovom slučaju je tekst, ali može biti i polje za potvrdu (checkbox), birač datuma (datepicker) itd. Za detaljan spisak tipova pogledati dokumentaciju HTTP protokola.

Pitanje

Koji od ovih atributa forma ne sadrži?

- method
- action
- **source**

Objašnjenje:

Tačan odgovor je da forma ne sadrži atribut source.

Django forme

Django forme predstavljaju spregu između HTML formi i Python funkcija i klasa. Logika za rad sa formama je smeštena u ugrađenu klasu forms, koja nam olakšava rad sa formama. Forme koje korisnik može praviti kroz Django radni okvir su po pravilu potklase klase forms (nasleđuje forms.Form klasu), a polja koja želimo da koristimo za unos podataka – njeni atributi. Takođe, prihvatanje podataka iz podnesene forme se vrši u views.py fajlu, dok se sama forma (ili više formi) može definisati u zasebnom fajlu. Za potrebe naše aplikacije za biblioteku, napravićemo formu sa dva polja preko kojih će korisnik unositi ime i godinu izdanja knjige, a pre definisanja takvog objekta treba kreirati i šablonski fajl sačinjen od HTML koda koji će pomoći Djangu u izradi i prikazu finalne stranice. Taj šablonski fajl ćemo nazvati form.html i smestiti u /templates folder u okviru naše aplikacije:

book_library/templates/forms.html

```
<form action="{% url 'book_added' %}" method="POST">
    {% csrf_token %}
    {{ form }}
    <p><input type="submit" value="Send the form!"></p>
</form>
```

U ovom kodu uviđamo da postoji samo jedan input element, koji predstavlja dugme za podnošenje forme. Ostali input elementi će biti kreirani nakon pozivanja render() funkcije u pogledu. Elementi Django šablonskog jezika u ovom primeru su:

- {% url 'book_added' %} – url predstavlja ugrađeni tag u šablonskom jeziku čiji je jedini obavezni argument zapravo putanja koja odgovara pogledu, odnosno – odgovara imenu koje zadajemo parametrom argumentom name u funkciji path() urls.py fajla (imenovanje samog URL-a). Takođe se ovom tagu mogu prosleđivati i argumenti, što se može videti u zvaničnoj dokumentaciji. Ono čemu služi atribut action elementa forme u ovom slučaju je doslovno: na link koji smo u urlpatterns listi imenovali kao book_added poslati podatke iz forme. Što znači da će nakon klika na dugme *Send the form!* pregledač otvoriti stranicu koja stoji iza naziva book_added (na taj link će poslati podatke iz forme). Iako možemo koristiti istu tu stranicu, koristićemo drugu, na kojoj ćemo kasnije ispisati poruku o uspešnosti poslate forme, tj. o ažuriranju liste sa knjigama.
- {% csrf_token %} – CSRF (cross-site request forgery) token koristi se za sigurnost aplikacije i potreban je prilikom slanja podataka na server radi validacije izvora

zahteva. Tačnije, server preko njega proverava da li zahtev dolazi sa domena sa kog Django i očekuje taj zahtev. Takođe, vezuje se za trenutnu sesiju. Koristi se najčešće prilikom POST zahteva klijenta. Više o toj temi se može naći na ovom linku i u narednim lekcijama.
- {{ form }} – mesto na koje ćemo proslediti objekat forme.

Nakon što smo definisali šablonski .html fajl koji će prikupiti podatke od korisnika u pregledaču, treba odrediti URL na kom će se taj fajl prikazivati i implementirati Django objekat tipa form koji će prihvatiti te podatke, kao i funkciju pogleda koja će ih procesirati.

Link na kojem želimo da se ovaj šablonski fajl pošalje će biti add-book/, a funkcija u pogledu, koju ćemo naknadno definisati, zvaće se get_book, pa tako u listu urlpatterns fajla book_library/urls.py dodajemo još jedan element, i to path('add-book/', views.get_book, name='book_added'),.

Pre definisanja funkcije treba prvo implementirati klasu koja će rukovoditi form.html fajlom. Takav fajl ćemo nazvati forms.py i smestiti u aplikacijski folder:

book_library/forms.py

```
from django import forms
class BookForm(forms.Form):
    book_title = forms.CharField(max_length=100)
    book_year = forms.CharField(max_length=4)
```

Iz ovog primera uviđamo da naša BookForm() klasa nasleđuje Djangovu forms klasu. U BookForm klasi smo definisali dva polja – book_title i book_year; oba pripadaju tipu forms.CharField(). Objekat CharField() zapravo predstavlja tekstualno polje koje će korisnik videti na ekranu. Tačnije, predstavljaće jedan od input HTML tagova. Kao parametar smo dodali i maksimalan broj karaktera koji može stati u to polje argumentom max_length. Django nam pruža dosta mogućnosti kada je reč o radu sa tagovima, pa tako podržava specijalizovane objekte za imejl (forms.EmailField()), datum (forms.DateField()), linkove (forms.URLField()) itd. Ostatak liste ovih polja koja Django nativno podržava se može naći u zvaničnoj dokumentaciji.

U primeru BookForm, za polja smo koristli max_length argument. Pored njega postoji još argumenata koji se takođe mogu primenjivati, kao što su:

- required – argument koji se odnosi na to da li je polje obavezno ili ne; podrazumevano je da je svako kreirano polje obavezno, ali ako želimo to da promenimo, treba proslediti required=False prilikom kreiranja polja. Primer: forms.CharField(max_length=100, required = False);
- label – argument koji nam omogućava dodavanje labele našem polju kako bi korisniku bilo jasnije na koji tačno podatak se samo polje odnosi; podrazumevana funkcionalnost ovog argumenta je uzimanje imena našeg polja, pretvaranje svih _ karaktera u razmake i pretvaranje prvog slova tog imena u veliko slovo; ako želimo drugačije ime, dovoljno je proslediti argument label i željeno ime; primer: forms.CharField(max_length=100, label = 'Name of the book');
- initial – argument koji se odnosi na to da li će polje sadržati neku inicijalnu vrednost ili ne; ovo je dobro u situacijama kada želimo da pokažemo korisniku kako treba da izgleda uneseni format podatka. Ako koristimo ovaj pristup, korisnik prvo mora obrisati tu inicijalnu vrednost pa uneti svoju.

Pošto u ovom trenutku imamo implementiranu klasu i definisan šablonski fajl, na redu je definisanje funkcija u pogledu. Treba takođe napomenuti da će tok aplikacije biti takav da će, nakon što korisnik pošalje podatke iz forme, ti podaci biti poslati na isti link na kojem se trenutno nalazi (127.0.0.1:8000/add-book), a da će se po uspešnosti ovog POST zahteva klijent preusmeriti na link 127.0.0.1:8000/book-added, na kome će pisati *Adding successful!*. Nakon toga, korisnik može videti unetu knjigu prilikom otvaranja adrese 127.0.0.1:8000/books. Zato, pre nego što implementiramo funkcije u pogledu, treba dodati još jedan element u listu urlpatterns fajla book_library/urls.py, koji sada izgleda ovako:

Izgled urlpatterns liste book_library/urls.py fajla:

```
urlpatterns = [
    # path('', admin.site.urls),
    path('',views.main_page, name = 'main_page'),
    path('books/', views.index, name = 'index_page'),
    path('books/<int:int_key>/', views.int_test, name = 'int_test'),
    path('books/<str:book_name>',views.index, name = 'str_test'),
    path('add-book/', views.get_book, name='book_added'),
    path('book-added/', views.redirect_test, name='redirect'),
]
```

Dakle, na redu je definisanje get_book() i redirect_test() funkcija. Nadovezujući se na kod iz prethodne nastavne jedinice, u book_library/views.py fajl dodajemo sledeći kod:

Dodatni kod za book_library/views.py

```
from django.http import HttpResponseRedirect
from .forms import BookForm
from django.urls import reverse
def redirect_test(request):
    return HttpResponse("<h1>Adding successful!</h1>")
 def get_book(request):
    if request.method == 'POST':
        form = BookForm(request.POST)
        if form.is_valid():
            books.append({'title':form.cleaned_data['book_title'],
'year':form.cleaned_data['book_year']})

            return HttpResponseRedirect(reverse('redirect'))
    else:
        form = BookForm()
        return render(request, 'form.html', {'form': form})
```

- odnosi se na argument name funkcije path());

U funkciji get_book() prvo proveravamo o kojoj metodi HTTP zahteva je reč. Naimje, ako je reč o POST metodi, znači da korisnik šalje podatke nama i moramo ih prihvatiti i uskladištiti. U tom slučaju prvo instanciramo našu formu – BookForm(), kojoj prosleđujemo podatke POST zahteva (request.POST – ova naredba je tipa rečnik). Bitno je da nakon toga proverimo ispravnost forme metodom is_valid(), koja je implementirana tako da se za sva podnesena polja koja su kreirana sa required argumentom koji ima vrednost True proverava da li sadrže podatke u sebi; ako sadrže, vraća True. U slučaju da je povratna vrednost is_valid()metode

False, korisnik bi ostao na istoj stranici sa svim podacima koje je popunio, a za polja koja su obavezna, a prazna izašlo bi obaveštenje. Pošto program utvrdi da je forma validna, nastavlja do linije kojom podnesene podatke ubacuje u rečnik (books.append()) i na kraju vraća korisniku odgovor sa kodom 302 – preusmerenje na link koji smo u urlpatterns listi nazvali redirect. Reč je o linku 127.0.0.1:8000/book-added/, za koji je zadužena redirect_test() funkcija koja ispisuje *Adding successful!*. Prilikom dodavanja, tačnije čitanja podataka iz forme, koristimo metodu cleaned_data, koja podatke iz forme pretvara u nativne Python tipove (int, str). Takođe, u ovom slučaju, pošto želimo da preusmerimo korisnika na stranicu sa potvrdnom porukom o dodavanju knjiga, koristimo HttpResponseRedirect umesto dosadašnjeg HttpResponse odgovora. U slučaju da zahtev koji dolazi do get_book() funkcije nije POST nego GET – znači da korisnik tek pristiže na stranicu na kojoj treba ispuniti formu i u tom slučaju mu se šalje form.html šablonski fajl.

Sačuvati sve fajlove i pokrenuti server komandom python manage.py runserrver '127.0.0.1:8000' i posmatrati odgovore servera na sledeće linkove:

- 127.0.0.1:8000
- 127.0.0.1:8000/books
- 127.0.0.1:8000/add-books
- 127.0.0.1:8000/books

Rezime

- Same forme su zapravo deo HTTP protokola.
- HTML tagovi koji nam služe za klijentsko slanje podataka serveru su form, input i select. Tag form je zapravo kontejnerski objekat za sve podatke koje klijent želi da pošalje u tom trenutku. Sadrži dva bitna atributa – action i method.
- Django forme predstavljaju spregu između HTML formi i Python funkcija i klasa. Logika za rad sa formama je smeštena u ugrađenu klasu forms koja nam olakšava rad sa formama.
- Za rad sa formama, pored .html fajla, moramo implementirati i klasu koja će rukovoditi tim fajlom. Takav fajl smeštamo u korenski direktorijum aplikacijskog foldera.